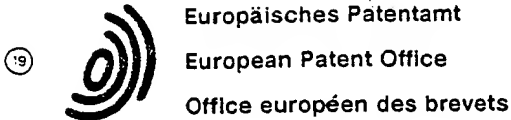


G06F 9/44

EP 10982 (5)

G06F 9/46 R 6C



(11) Publication number:

0 369 961
A2

(12)

EUROPEAN PATENT APPLICATION

(21) Application number: 89850374.3

(51) Int. Cl. 5: **G06F 9/46**

(22) Date of filing: 31.10.89

(30) Priority: 18.11.88 US 273527

(43) Date of publication of application:
23.05.90 Bulletin 90/21(54) Designated Contracting States:
DE FR GB(71) Applicant: **International Business Machines Corporation**
Old Orchard Road
Armonk, N.Y. 10504(US)(72) Inventor: **Chang, Phillip Yen-Tang**
6221 Ledge Mountain Drive
Austin Texas 78731(US)
Inventor: **Ronga, Gerald Samuel**
9902 Richelleu
Austin Texas 78750(US)(74) Representative: **Johansson, Lars E. et al**
IBM Svenska AB Intellectual Property
Department 4-01
S-163 92 Stockholm(SE)

DOC

(54) **Interface device and method in a computer system.**

(57) A linking system for use in a computer environment which supports interapplication messaging and interactive dialog. A link protocol comprised of a set of messages is provided and employed by a link service to interrelate information existing in single or multiple applications. This open protocol facilitates non-joint development of application programs for use in the system. The link service modules provide end user function permitting tasks executed dynamically at run time, of creating or saving links between two objects either of the same or different object types and with or without data exchange, the creation or saving of an application packet comprised of one or more links, refreshing by warm or hot links any linked object, and displaying and navigating links of a packet causing objects to be retrieved and/or refreshed. Function is provided for limited participation in the system by application programs preexisting prior to the protocol.

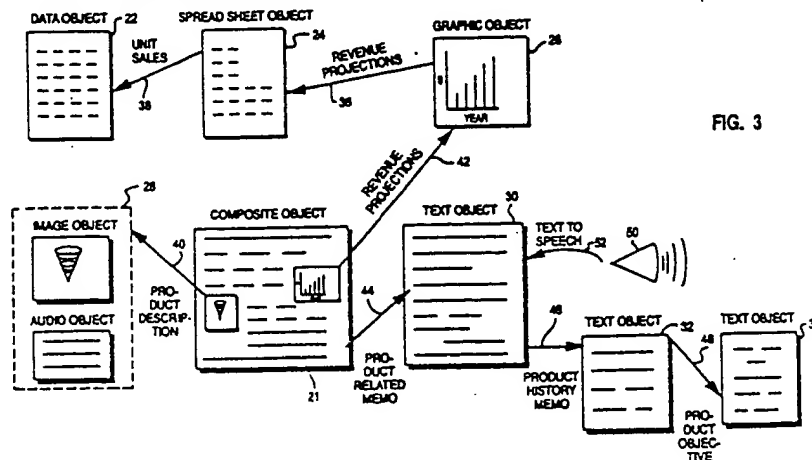


FIG. 3

EP 0 369 961 A2

data associated with a plurality of cells) into a graphic form which may be accomplished by the graphics package 12 as, for example, in the bar graph depicted therein.

In effecting such desired results several characteristics of these application integration systems may be noted. First, these are closed environments or systems, i.e., only applications built within the environment with detailed knowledge thereof may participate. It will be recalled that this requires the application programmer to learn details of the system such as private internal calls which are conventionally not published, details regarding the shared memory segments of the environment, and the like. This in turn obviously requires a great deal of joint development time and interchange between developers of various applications desired to be executed in the system.

Moreover with reference to Fig. 1, it will be noted that only temporary data links 13 are conventionally established between applications. Therefore an object from one application may be transferred to the second application but the interrelationship therebetween is not saved beyond this current instance of either application, i.e., it is destroyed whenever either of the applications involved is terminated. It follows from this that no link navigation support is provided whereby either during initial run time or thereafter these links might be recalled for inspection, editing, or the like. Again, it will be recalled that such data link type exchange had to be done through internal calls of the particular application which were typically private and not published, requiring detailed knowledge of complicated internal data structures, and the like thereby hampering a more universal integration of applications outside the closed environment. Related to this of course were the technologies for straightforward data transfer between dissimilar systems, again typically involving nothing more than format conversion and the like to facilitate exporting data between two applications for example.

With reference to Fig. 2 a schematic illustration is provided of the characteristics of the other major closed environment system wherein the benefits of linking of information is sought to be obtained, namely the hypertext/hypermedia systems currently emerging and commercially represented by the products Guide (Owl International), Hypercard (Apple Computer Company), and a research project known as Intermedia (Institute for Research in Information and Scholarship, Brown University). As distinguished from the application integration systems of Fig. 1, in these systems of Fig. 2 collectively referred to for simplicity as hypersystems, information existing in applications is only interrelated or linked (as opposed to the data actually being transmitted in the case of Fig. 1). Thus, referring to Fig. 2 objects such as information contained in a bar graph 14 may be related to a sentence in a text document 16, which may describe the bar graph 14 in some way. Further a word 17 in the text document 16 may be further explained by a paragraph in text document 18 depicted therein.

Once again it is important to note that these hypersystems are closed in the sense that only applications built within the particular hypersystem environment (i.e., using a specific language and/or set of development tools) may participate. In the operation of such systems, it is typical to find capability to establish persistent reference links amongst various objects or subobjects as well as link navigation support. In this manner, for example, the object of the bar graph 14 might be interrelated to a text document 16 by user interaction with the hypersystem whereby their logical interrelationship indicated by reference link 20 is established and retained. Thereafter, the user might desire to and be capable of referencing the bar graph data 14 from the text document 16 by recalling the links previously established.

Whereas persistent links and link navigation support may be provided, unlike the application integration systems of Fig. 1 these hypersystems typically nevertheless suffer from the same serious drawback of the Fig. 1 systems, namely that they may be found only in closed systems with all the attendant serious problems associated therewith. Moreover, typically in the hypersystems of Fig. 2 unlike the Fig. 1 systems, only reference links are provided for and not data links, whereas in the Fig. 1 systems only data links are provided for and not reference links.

Another system which sought to provide the benefits associated with both the hyper and application integration systems was a system known as Metaphor provided by the Metaphor Computer Systems Corporation. Such a system did provide for data links and a means for storing and navigating link data but nevertheless still suffered from numerous drawbacks. First and most importantly, the system still remained closed in the sense that no other applications could participate in the linking functions without detailed knowledge of the Metaphor environment. Also, reference links were not supported. With respect to provision of link data storage, there was no provision for dynamically storing at run time during a session link data which was being created at run time during the session. In contrast, after a user session with the product, a separate procedure was required for defining permanent links specified and moreover even after such a separate procedure for effecting persistent links, these links were not hot. Thus these deficiencies seriously affected productivity.

Given the foregoing deficiencies of the closed systems of Figs. 1 and 2, efforts were made in the art to

Fig. 9 is a more detailed flow diagram of the logic for handling link user interaction shown in block 122 of Fig. 8.

Fig. 10 is a more detailed flow diagram of the logic for handling link protocols as shown in block 124 of Fig. 8.

5 Fig. 11 is a detailed flow diagram of the logic for link packet manager (LPM) functions provided by the link packet manager of Fig. 4.

Figs. 12A-G are detailed functional diagrams of the functions performed by the link message manager (LMM) of Fig. 4 after receipt of protocols from an application C or link packet manager of Fig. 4.

10 Fig. 13 is a diagram of the flow used by the service to receive and send data between two LP applications.

Best Mode For Carrying Out The Invention

15 With the characteristics of prior systems in mind which were hereinbefore described with reference to Figs. 1 and 2, Fig. 3 depicts pictorially the link service environment of the present invention and the distinctions over these prior systems. First it will be noted that the present invention provides for composite objects 21 to which may be interrelated a plurality of other objects or subobjects of widely varying types by means of links which may also vary in type and function. For example, the composite object 21 may have
20 image or audio objects 28, text objects 30, or graphic objects 26 interrelated by links 40, 44, and 42, respectively. The graphic objects 26 may further be interrelated to still other objects such as a spreadsheet object 24 or data object 22 by means of links 36 and 38. Similarly, the text object 30 may in turn be related to yet further text objects 32 and 34 by links 46 and 48, respectively. To illustrate that the invention contemplates interrelating widely differing types of objects, a text to speech object 50 has further been
25 included and is interrelated to text object 30 by a link 52.

Not only does Fig. 3 illustrate the wide variety of objects interrelated in accordance with the invention, but the notion of both data and reference links being provided for in the same system. For example, when users are interacting with composite object 20 it may be desirable to simply note by means of link 40 the previously established interrelationship to image or audio objects 28 thereby referencing the information
30 contained in these objects 28. In this case, the link 40 is merely a reference link and such links as well as the objects they interrelate may be graphically depicted to the user so as to enable editing, navigating or establishment of yet additional links as desired.

Similarly, a reference link 42 may be established whereby it is apparent from inspection of the composite object 20 that interrelated data is available for inspection, editing, and the like by reference to
35 graphic object 26 established through the reference link 42. However, link 36 is intended to indicate a feature of the invention wherein data links such as 36 are also provided which may be established, navigated and edited as required by the user. With respect to these types of links, changes made in an object such as cells in the spreadsheet object 24 may be reflected, either automatically or semiautomatically in another object such as the graphic object 26. Thus, for example, changing data referring to cells in the
40 spreadsheet object 24 would automatically be reflected as changes in the graph of the graphic object 26. As will be made clearer hereinafter, when these data links such as link 36 are established, as an alternative to their functioning as hot links wherein these automatic object changes transpire, this function may be specified by the user to be semiautomatic or "warm". In this manner, the user may interact in the link service environment by creating or editing objects, establishing or modifying data links, changing data and
45 the like during a session without changes to a particular object or subobject manifesting themselves in related objects or subobjects until an affirmative desire to do so is indicated by the user.

Still referring to Fig. 3 yet additional features of the invention will be herein described in general terms to facilitate a more detailed description of the operation and function of the invention which follows. The illustrative collection of objects or subobjects and links as shown in Fig. 3 may be referred to as a "packet".
50 In accordance with the invention a user may desire a logical break for differing sets of such links and objects which may be broken up or organized arbitrarily by any of a plurality of factors. For example, the user may establish a number of such packets distinguished in the sense that each is created for a different function. It is integral to the invention that all such packets are saved in permanent storage for subsequent recall and interaction which might include navigating the links for data retrieval, editing thereof to establish
55 new logical relationships or changing data which may effect changes in other related objects or subobjects in the case of the aforementioned data links.

In Fig. 4 there may be seen a functional block diagram depicting the link service architecture or module of the present invention. First the overall functions of major components thereof will be described followed

the user for link types and link attributes in response to which the user would specify through the I/O link between the user 64 and the link service of the invention the requested link type and link attributes. These are provided back to the LS by means of appropriate protocol messages of Table 1.

In response to the these received link type and attribute specifying messages, the LS performs the function of format negotiating between the applications 82 and 88 and initiates the start of data exchange wherein, for example, the first application 82 provides data which is received by the second application 88. The LS will thence display an indication of the link and its attribute in the dialog box 84. This might include a graphical line 87 interconnecting the applications 82 and 88 which might be highlighted or otherwise indicated for example in window 90 as signifying that the link just created was a hot or warm data link.

Fig. 6 is intended to indicate the function contemplated by the invention wherein the link service is intended to provide at least limited support to existing applications. Now with reference to Fig. 6, an existing application 92 may be used as the start or end of a link. Such application 92 must support only one characteristic generally satisfied by many applications existing in the market place, namely the support for an invocation which allows an object or subobject to be referenced and displayed as part of the invocation. The participation by application 92 is limited to reference links, and the user 100 must interface through the link service 98 directly to create the spreadsheet 92 portion of the link. This interaction is provided by the link packet manager LPM 60 of Fig. 4 and is shown as functional interconnection 102 of Fig. 6. In Fig. 6, the existing application 92, such as a spreadsheet, may be an endlink for a text application 94 as depicted by link 106 or a startlink for an audio application 96 as indicated by link 104. In this example since both applications 94 and 96 support the link service protocol, the user 100 may reference the spreadsheet 92 from either application subobject during execution. The only way however, to reference either application 94 or 96 from the spreadsheet is through the use of the graphical representation provided by the LPM 60 of Fig. 4. Applications 94 or 96 may not be referenced directly from application 92 during execution. Neither application 94 nor 96 has to support the protocol for a link to be created as indicated in Fig. 6, however, in that case the restrictions discussed above would apply to both sides of the link.

In Fig. 7 a functional block diagram illustrates apparatus for implementing the environment of the invention. In the computer system environment 101 depicted therein, a microprocessor 103 intercommunicates by means of a bus 105 with a plurality of adapters 107-117. The bus 105 is only illustrated generally and intended to represent any conventional bus structure comprises of data, control, and address signals for effecting the intercommunication between the microprocessor 103 and the adapters 107-117. Each adapter has associated interconnected therewith a device such as those shown at reference numerals 119-129. The input devices 119 block is intended to represent any number of such devices well known in the art for receiving input from a user 131 such as a keyboard, mouse, touch screen, or the like. The display 121 is intended in the conventional sense for conveying graphical images such as those represented generally in the other figures to the user 131 such as displays of packets, objects, subobjects and their links either as they are selected or edited by the user 131 or as retrieved in accordance with the invention. Storage media 123 is intended to represent any bulk storage media as desired which may take the form of disk or tape drives, or the like. This media 123 will be used to permanently store the links described previously. RAM memory 125 is further provided in the system 101 for conventional purposes and will be used by the invention to cache the packets. ROM memory 127 is also included which may take the form of CD ROM or the like. Block 129 is intended to indicate that other output devices may be included in the system 101 for implementing the invention's function as hereindescribed such as a printer for hard copy of any I/O between the system 101 and the user 131.

The system is further provided with an operating system 137 which (under control of the microprocessor 103 through the bus 105 and respective storage media 123-adaptor 111 and RAM 125-adaptor 113 pairs) performs the functions normally associated with conventional operating systems as well as including function required herein with respect to the link services as well as the link service itself. The functions which the time service is dependent upon include a file system or database manager, a dialog facility and messaging facility. The file system or database manager is used by the invention to facilitate the storing of links/packets. The system 101 is intended to operate with one or more applications 133 which may or may not be object oriented programs and may be either link protocol type programs to participate fully in the benefits of the invention or, in the alternative, may be non-link protocol applications. As to the later and as will be hereinafter described in more detail, even such non-link protocol applications may at least enjoy some limited participation in the environment.

The architecture and function described herein relative to the invention and depicted as the generalized view of Fig. 7 may be implemented in terms of hardware support by a number of computer systems such as the configuration used in the IBM Personal System/2 family of computers which may include an Intel 80286 or 80386 or similar microprocessor 103. The display 121 in such a system might for illustrative

a spreadsheet, the user would identify an area of the spreadsheet by name which would be sent upon start of the link to the LMM at 138.

Once the application has sent a call to the LMM at 119 of Fig. 8 and thereby obtained the information associated with the object, it is then able to transmit to the LMM (at 138 of Fig. 9) the start link message including the active packet name. For each action 136-156 of Fig. 4 relating to the user interaction, the user action selected will be sent along with the packet name.

In summary then, in response to a received user action at 134 (to start link 136, end link 140, follow link 144, refresh link 148, or delete link 152, various protocol information as indicated in the Table 1 corresponding to the associated indicated user action will be sent to the LMM (start link protocol data 138 in response to start link user action 136, end link protocol data 142 in response to end link user action 140, follow link protocol data 146 in response to follow link user action 144, request data protocol information 150 in response to refresh link user action 148, and delete link protocol data 154 in response to delete link user action 152.

With respect to the specific user actions 136-152 of Fig. 9, further detail as to the nature of these actions is helpful. A link is created by the user actions of a "start link" 136 and "end link" 140. However, with respect to the "follow link" 144 indicated user action, this refers to the LP application function wherein the particular subobject in the application is selected and it is desired to follow an existing link to the other objects and subobjects to which the selected subobject is linked. The "follow link" results in the linked object and subobject being invoked and displayed.

In response to the protocol sent at 146 relating to the "follow link", the LS handles all follow link actions itself. In this case the applications only responsibility is to indicate the user action by sending the "follow link" message.

With respect to the "refresh link" user action 148, this refers to situations in an LP application wherein a subobject in the application was linked to a subobject in another such application and the link was designated as "warm". A "warm" link, it will be recalled, means that the subobject does not automatically flow between applications when the data changes, in contrast to hot links previously discussed. When a user is looking at the end of a link and desires to see up-to-date data relating to a specific subobject, this is when the refresh link user command 148 comes into play. In response to such a "refresh link" indicated user action, the LP application sends the "request data" protocol message to the link message manager (LMM) which in response thereto effects such refreshing.

When a user desires to delete a link as indicated by the "delete link" user action 152, the application sends a "delete link" message in the corresponding protocol defined in Table 1 to the LS as shown at 154.

As indicated from the looping out from 152-154 back to the entry point 134, this process is continuously repeated of repetitively scanning for all indicated user actions 136-152 and handling thereof by way of the application sending protocol messages 138-154 to the LMM.

In summary then, with respect to the user actions, it should be apparent that the "start" and "end" link actions 136 and 140 are associated with creating a link. The "follow" and "refresh" link actions 144 and 148, respectively, relate to use of the existing links, and the "delete" link action 152 indicates a desire to change a link.

With reference now to Fig. 10, the detailed flow diagram indicates the manner in which an LP application processes link protocols as shown generally at 124 of Fig. 8. It will be recalled from previous discussion with reference to Table 1 that such protocols refer to information sent either from the LS to the LP application or vice versa and that the LP application must be designed so as to handle such protocols. As indicated at 160, the link message manager (LMM) will send LP messages to the LP application. The protocol includes specific LP application IDs (indicated as "object type" in the parameter list of the protocol messages in Table 1) which identify whether such messages are intended for the particular LP application. As a result, once the application has thereby determined that particular link messages are intended for the particular LP application, the application will first determine what the message is inasmuch as there are a number of possible messages such as those indicated in Fig. 10 as 162-190.

One message which the LP application scans for is whether there are errors returning from the link service as indicated by the error decision block 162. If the application determines that errors exist, the application would return the indications of the particular errors to the user at 164 enabling the user to take appropriate corrective action. Such errors typically might include indications of incorrect parameters or non-existence of links for example.

Still referring to Fig. 10, it will be noted in passing that the particular messages 162-190 being depicted in the process of Fig. 10 are being sent to the application by the LMM, while the LMM is performing the specific functions more particularly described hereinafter with reference to Figs. 12A-G. Thus, for example, the "error" being sent to the application at 144 in Fig. 12A by the LMM corresponds to the error received

specific actions which the user desires to take with respect to the selected subobjects and links. Such actions include those specifically shown with reference blocks 96, 110, 112, 118, 124, and 130 of Fig. 11.

When the LPM detects at 96 that the user action has indicated a desire to select a packet, the LPM generates a message at 98 querying the user for the desired packet name. Upon the user specifying a desired packet name, the LPM determines at 100 whether the specified packet name corresponds to an existing packet. If the packet is not already in the system, the LPM will create a new packet at 102. If the packet is already existing in the system, the LPM will retrieve and cause the caching of the packet at 104. In some cases, a user may wish to view a packet. The "show packet" user action, indicated by 110 in Fig. 11 provides such a function.

In some instances a packet may be rather large with numerous subobjects and links. In such instances it may be desirable to provide a more simplified indication of the overall link structure or a more detailed view of a particular portion of the links the user is interested in. Thus, the LPM will accept as indicated at 108 user provided keywords to filter links. The LPM then attends to generating commands so as to cause the display of the links having the key words only, 106. Thus, in a packet having numerous links, only those links are displayed satisfying the key words so the user may concentrate on them and determine if they are desired links, thereby simplifying the graphical packet display.

It is also desirable to enable a user to start several links without requiring the user to start and end one link at a time. For example, a user may desire to go to one or more applications and start more than one link and subsequently complete a particular link from start to end. The steps 112-116 provide a convenient means for doing so. When the LPM detects a user action at 112 indicating a desire to show all start links, the LPM generates necessary messages to cause the display of all such start links together with their correlative objects and subobjects at 114. At 116, the LPM queries the user to indicate whether it is desired to delete a start link and/or specify a displayed start link as being current. There are several advantages to the packet manager providing these functions. For example, if one start link is made current several end links may be specified whereby one start link may be linked to multiple end links very readily. Moreover, with several start links, different ones can be made current at 116 at different times so as to associate them with the desired specified end links.

An alternative to invoking this function explicitly is the case in which the user has indicated multiple start links and then chooses an end link. In this case the list of start links will be displayed automatically and this function invoked implicitly. Also, if only one start link exists when an end link is selected this function will be bypassed.

In Fig. 9 it will be recalled that functions are specified at 136 and 140 to be performed by the LP application in starting and ending links. In contrast, with reference to Fig. 11, these start and end link functions 118 and 124 are specified by the user through the link packet manager (LPM) and are accordingly LPM functions. The significance of this is that for non-LP applications, such as that schematically indicated in Fig. 6, it is the LPM rather than the application which will support these functions specified by the user in starting and ending links at 118 and 124 of Fig. 11, respectively. Thus, for "start link", when the LPM determines at 118 that this is the desired specified user action, it generates the appropriate messages requesting specification from the user of a desired object and subobjects at 120 whereupon the information provided by the user in response thereto is sent to the LMM as indicated at 122. In like manner, once the LPM determines an end link is desired at 124 in response to the user action 94, messages at 126 to the user request specification of the object and subobjects, and user responses thereto are also sent to the link message manager (LMM) as indicated in Fig. 4 at 128.

Yet another user action 94 which is detected by the LPM at 130 is the selection of an object from the graphical packet displayed on the monitor. In this case, as indicated by the options 132, the user may provide user actions in response to visual user prompts to start a link, 133, end a link, 135, delete a link, 137, or show an object, 139. Again there is a distinction between these start and end link actions 133 and 135, respectively, and the start and end link steps 118 and 124, respectively. Link actions indicated in 118 and 124 are used for non-LP applications the first time a specific object or subobject is used in a link. However, for an object or subobject that already exists in a packet it may be more convenient for the user to interact directly with the graphical packet representation. Such actions are indicated by steps 133 and 135. Whether an LP or a non-LP application, the object would appear in the graphical packet displayed on the monitor so that the user may cursor to the particular object on a visual display and select it at 130. The option 132 thus indicates that upon the user action indicating a desire to start, end, or delete a link, 133-137, a correlative message 134, 136, or 138 is sent to the LMM to start, end, or delete a link, respectively.

With respect to the user action of indicating to the LPM a desire to show an object, 139, the action taken by the LPM is to send a call to the application to start it with the particular designated object and subobject as indicated at 140. If there are multiple links associated with a particular object, the first link into

establishment of the link. The DLS will also send a message (as indicated at 192) to both applications acknowledging that the completed link has been established. Each application will then handle this message as indicated at 168 Fig. 10.

With reference now to Fig. 12C, this flow diagram indicates how the LMM responds to a "follow link" message 194. This series of functions shown in Fig. 12C performed by the link service is in response to the "follow link" message sent by the LP application as represented at 146 in Fig. 9. Basically, the LMM first checks to see if there is more than one link, 196. If there is more than one link, the LPM queries the user to specify which link to follow, 198, either by specifying one or more key words or by designating a link from a graphical display of links by means of a pointing device or the like. As represented at step 200, the LMM determines whether the object's destination or end of the link is active at that point. If it is, the link service will send a "show object" message to the application, 204, causing the application to display the subobject. If the destination is not active, the application is invoked by the LMM at 206. This call will start the application with the object and subobject pre-specified at 116 in Fig. 8.

With reference to Fig. 12D, this flow diagram indicates the functions performed by the LMM in response to a "request data" message sent to the LMM at 150 of Fig. 9 by the LP application. Upon receipt of the request data message at 210 in Fig. 12D, the LMM determines at 212 whether the LP application which owns the subobject at the start of the link is active. The reason for this is that if the source is active, all the LMM needs to do is pass the request for data along to the LP application containing this subobject where upon the LP application will provide the data to the LMM which will then pass it to the requesting LP application as previously described. Thus, at 214, if the source data is active, the LMM sends a message to the application containing the subobject (data) requested. This data request, it may be recalled, is handled by the LP application at 174 of Fig. 10. If the source is not active, the LMM will invoke the application with the object and subobject parameters at 216 and then send the "request data" to the application at 218.

At Fig. 12E, this flow diagram indicates the functions performed by the LMM after receipt of a "data changed" message 222 from the LP application. Such message, it will be recalled, was generated at step 132 of Fig. 8. Specifically in handling this "data changed" message, the LMM first determines whether any destination is active at 226. This situation differs from the previously described determination at 212 of Fig. 12D in handling the "request data" message. When data is changed in a subobject with a hot link and the destination application is not active, there is no need to transfer the data to this application and show it. Thus, at 226 the determination made by the LMM is whether the destination is active whereby, if active, it performs at 224 the same data exchange for all hot links as shown in Fig. 13. However, if the destination is not active as determined at 226, then the LMM subroutine of Fig. 12E ends at 225. Referring briefly back to step 224, it will be noted that the data exchange referenced at that step is for all hot links and thus, for example if three hot links exist, the functions outlined with reference to Fig. 13 will occur for each such link. The reason for this is that if data is changing in a subobject associated with hot links, this may cause several data exchanges to occur with respect to these multiple links.

With reference to Fig. 12F, this flow diagram indicates the functions performed by the LMM in response to message 248 of previously discussed Fig. 13. In this case data is being sent from an application 242 of Fig. 13 to the LMM 252 of Fig. 12F in response to an original request data message 246 of Fig. 13 sent to the application 242 by the LMM. The LMM must now determine in box 254 of Fig. 12F, from prior user end link interaction discussed previously, which application to send the data to. In box 256 of Fig. 12F the actual data message is sent to the appropriate application as illustrated in box 250 of Fig. 13.

With reference to Fig. 12G this flow diagram indicates the functions performed by the LMM in response to a message 154 of Fig. 9 from an LP application indicating a desire to delete a link. Upon receipt of the "delete link" message 228, the LMM will delete the link from the cache and from the database, 230, and send to the applications on both ends of the link or links at 234 messages causing the unmarking of the subobjects related to the links. This message 234 from the LMM to unmark subobjects is handled by the LP application function at 186 of Fig. 10. Fig. 13 has been described previously with respect to step 182 of Fig. 12B.

Table 1 which follows provides a representative listing of the link protocol messages required to effect the objects of the invention which are either provided by the application or applications 133 or the link service of the invention. Following each message name such as "START_LINK" is the list (in parenthesis) of the parameters required to define the particular message which may either be provided by user 131 input to the system 101 of Fig. 7, the application 133 or may already have been provided, stored, and retrieved as necessary by the link service. Comparison of the messages of Table 1 with the flow diagrams of Figs. 8-13 will reveal the logical relationship between the respective messages and the component or function generating the particular message or receiving it. For example, in Fig. 7, again using the "START_LINK" messages as an example, this message will be seen to occur at block 136 in Fig. 9.

ERROR(OBJECT_TYPE, OBJECT_NAME, ERROR_CODE, ERROR_MSG,
ERROR_INFO)

DATA_CHANGED(PACKET-NAME, OBJECT-TYPE, OBJECT-NAME,
SUB-OBJECT-NAME)

Claims

1. An interface for use in a computer system environment between one or more application programs comprising a link service code module means responsive to a set of open link protocol messages for interrelating objects or subobjects of said one or more programs.
2. The interface of Claim 1 wherein said module means includes means for generating, in response to at least one message from said set of protocol messages, a reference link interrelating objects or subobjects defined by said one or more application programs.
3. The interface of Claim 2 wherein said module means includes means for generating in response to at least one message from said set of messages a data link interrelating objects or subobjects defined by said one or more application programs.
4. The interface of Claim 3 wherein said module means includes means for generating in response to at least one message from said set of messages, commands for storing said reference or said data links.
5. The interface of Claim 4 wherein said data links are warm or hot.
6. The interface of Claim 1 wherein said application programs comprise first and second application programs each supporting respectively at least one information exchange format, and wherein said code module means further comprises format negotiating means for selecting, in response to open link protocol messages transferred between said code module and said first and second programs, a common one of said information exchange formats for information transfer between said first and second programs.
7. The interface of Claim 6 wherein at least one of said application programs is a non-link protocol application program.
8. The interface of Claim 7 wherein said transferred information is between link protocol applications.
9. The interface of Claim 8 wherein said code module means includes cache memory means for storing said packet or link data.
10. The interface of Claim 9 wherein said code module means further includes database interface means for allocating and managing said cache of packet or link data.
11. The interface of Claim 10 further including link message manager means for analyzing said packet or link data to provide graphical representation of said data.
12. A method for use in a computer system environment to interface one or more application programs comprising receiving in a link service code module at least one open link protocol messages from a first of said one or more application programs defining a link relationship between objects or subobjects of said one or more programs; and generating in said link service code module in response to said received at least one protocol messages a command corresponding to said link relationship between said objects or subobjects for storing said link relationship.
13. The method of Claim 12 further including storing said link relationship in a persistent storage media in response to said command generated by said link service code module.
14. The method of Claim 13 further including generating in said link service code module a next command; and retrieving said stored link relationship from said persistent storage media in said link service code module in response to said next command.
15. The method of Claim 14 including generating in said link service code module, in response to said retrieved link relationship, protocol messages corresponding to said retrieved link relationship; and invoking a second of said one or more application programs with said protocol messages generated in response to said protocol messages generated in response to said retrieved link relationship.

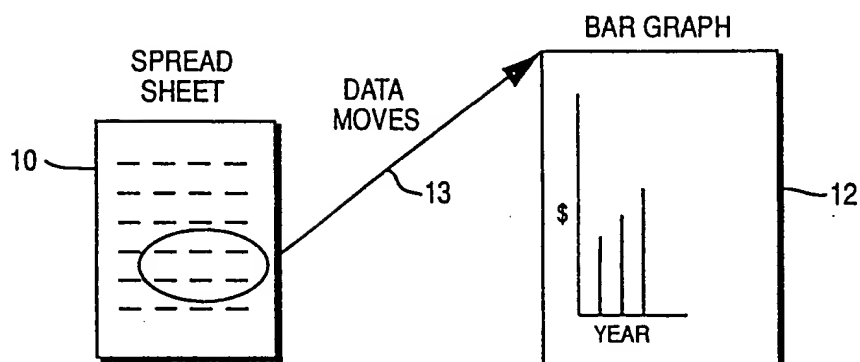


FIG. 1

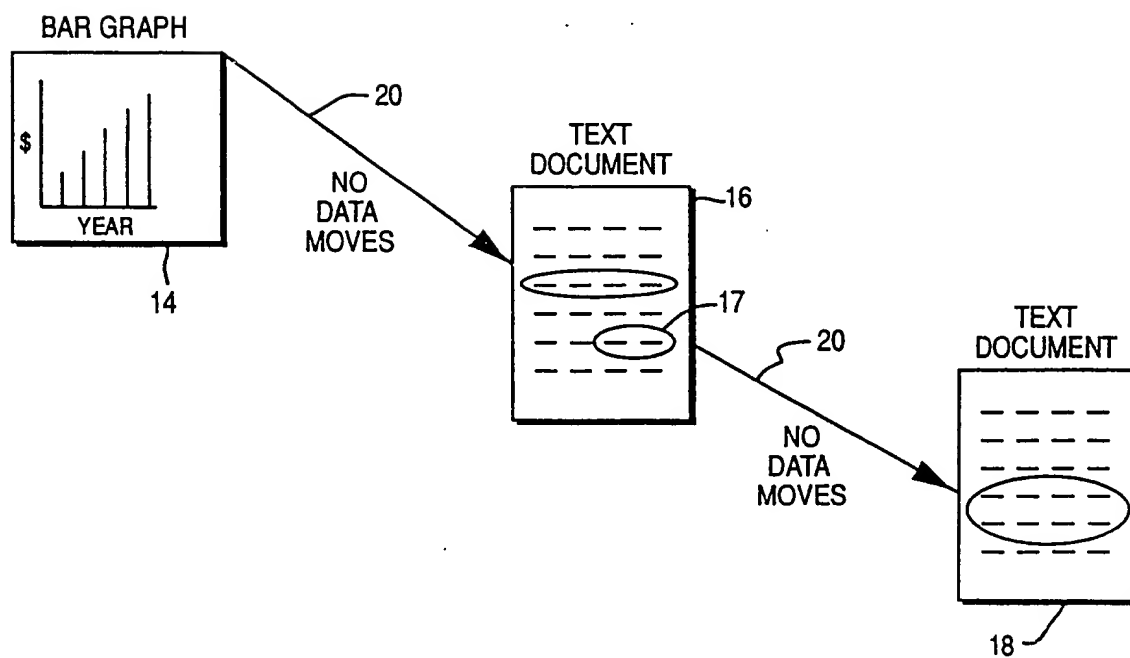
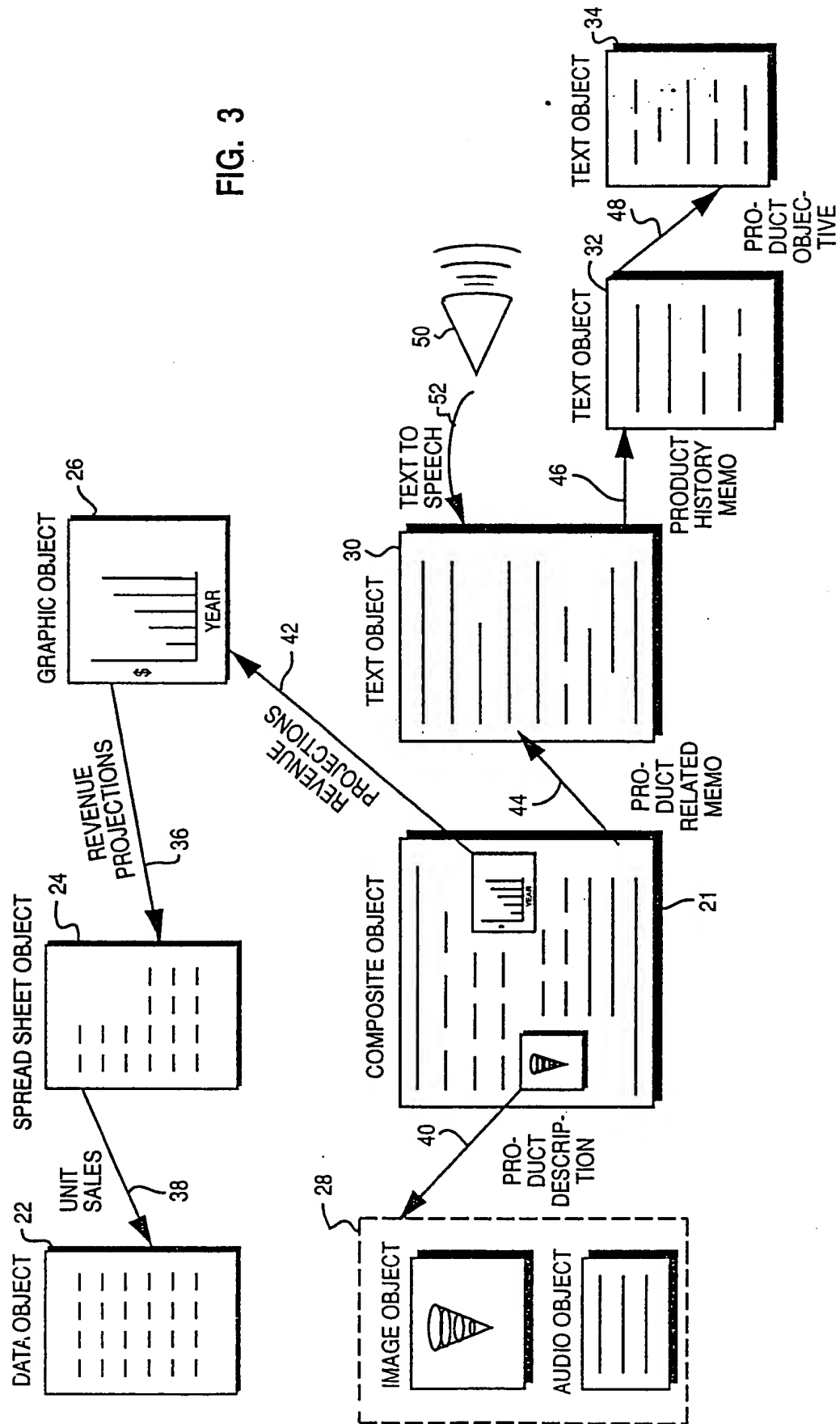


FIG. 2

FIG. 3



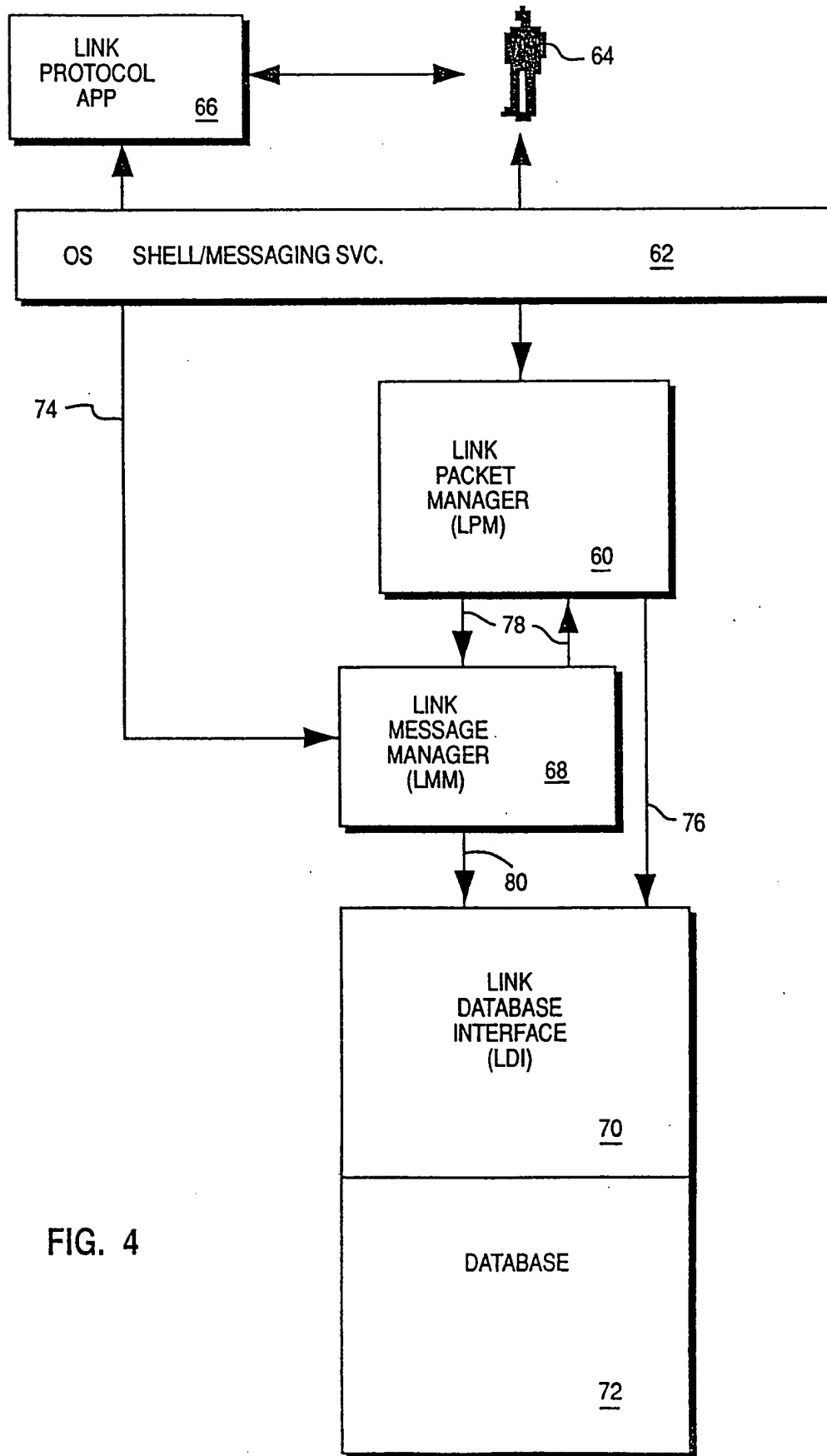


FIG. 4

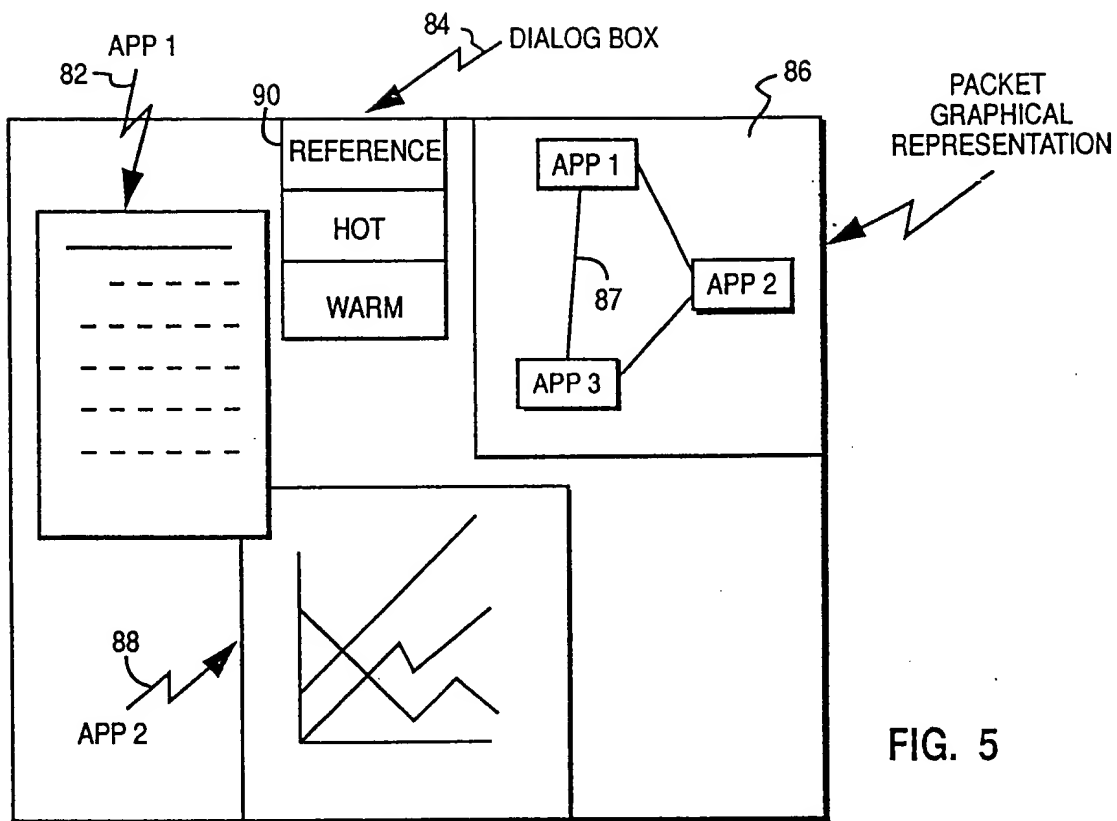


FIG. 5

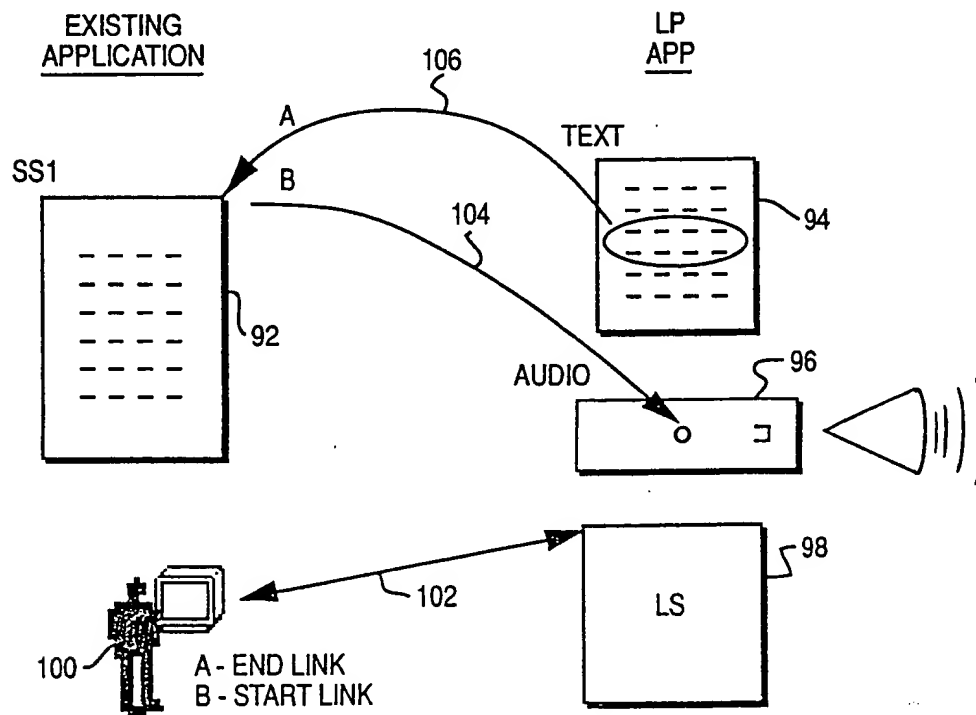
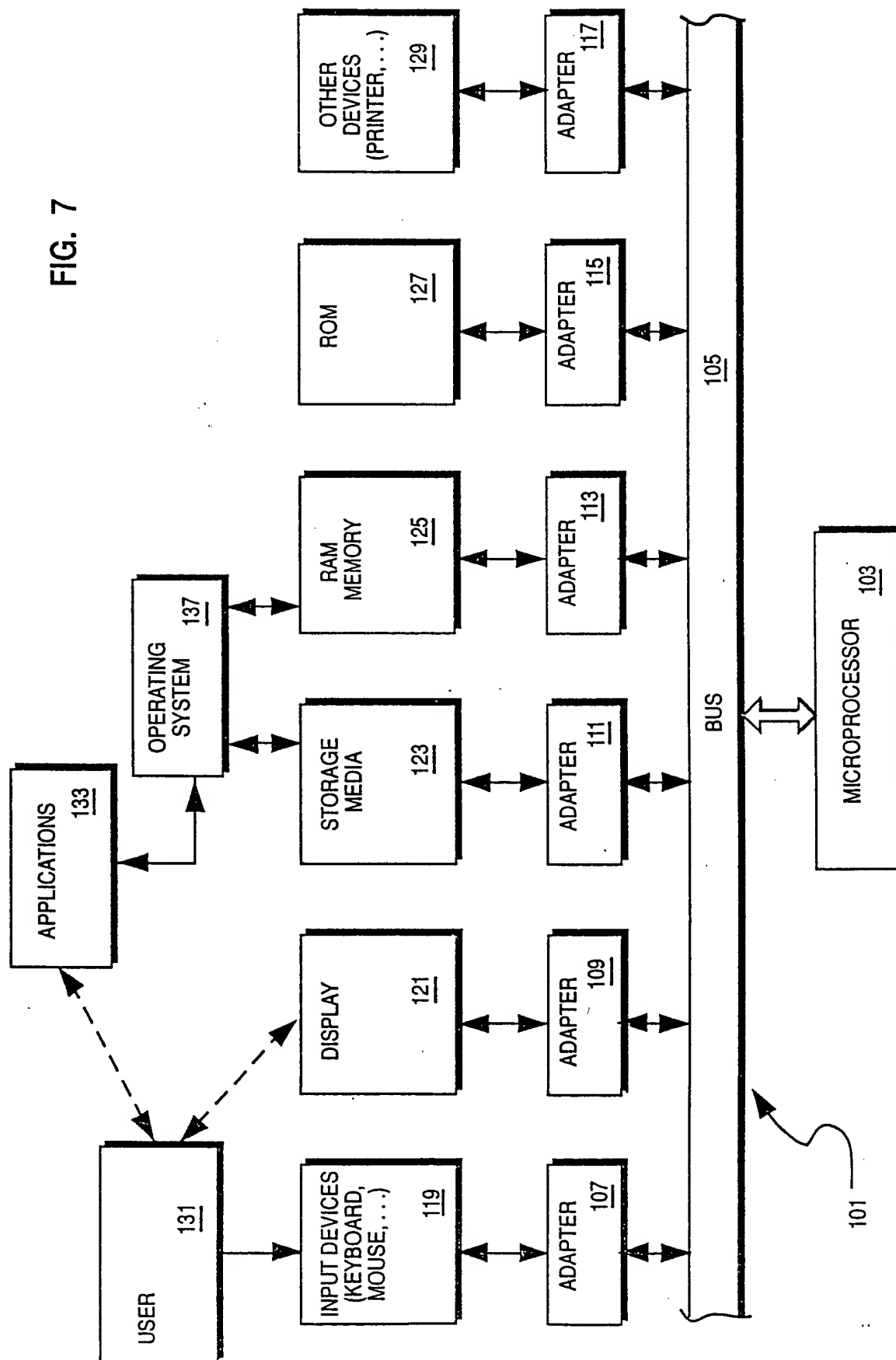


FIG. 6

FIG. 7



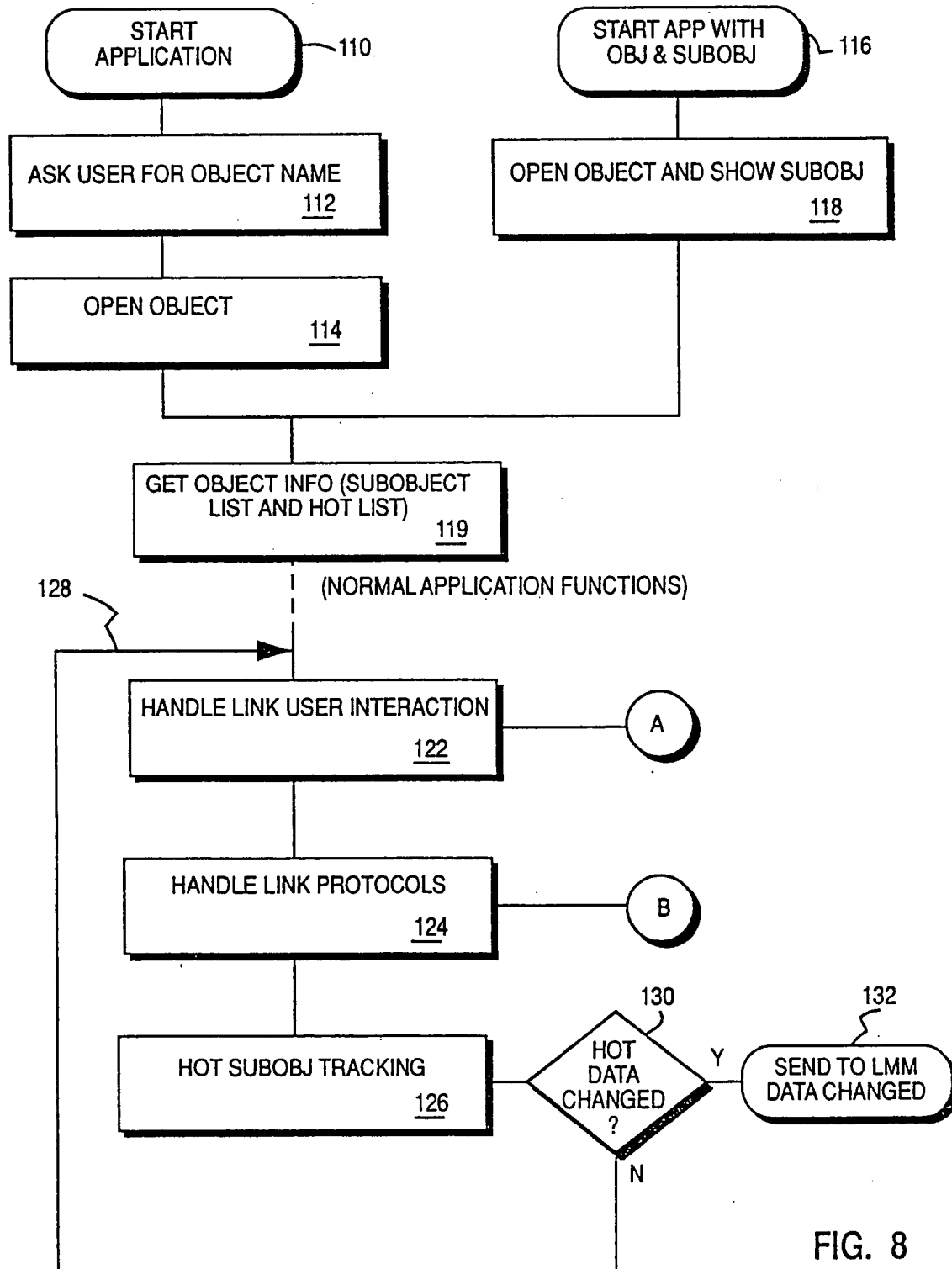
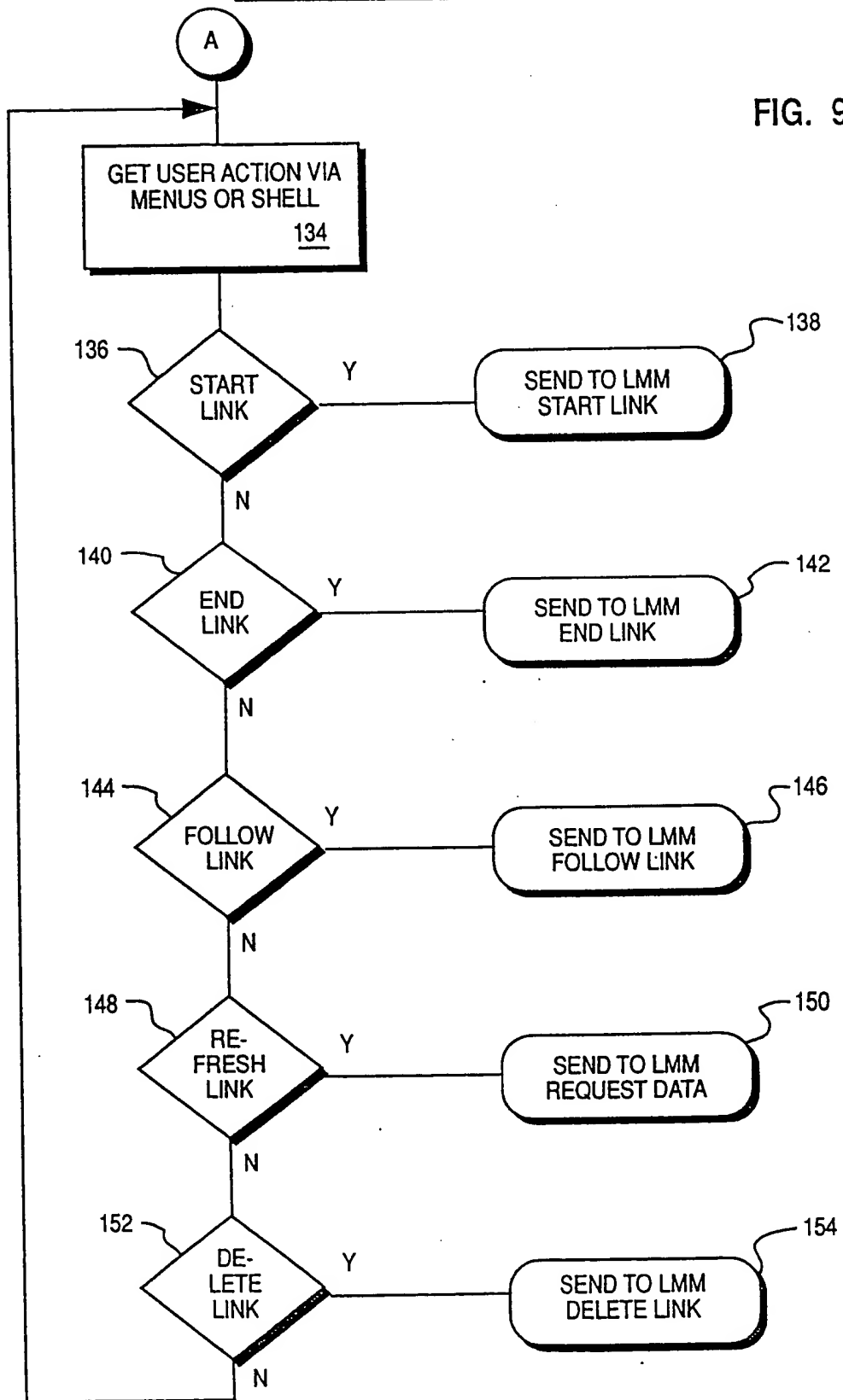
LP APPLICATION

FIG. 8

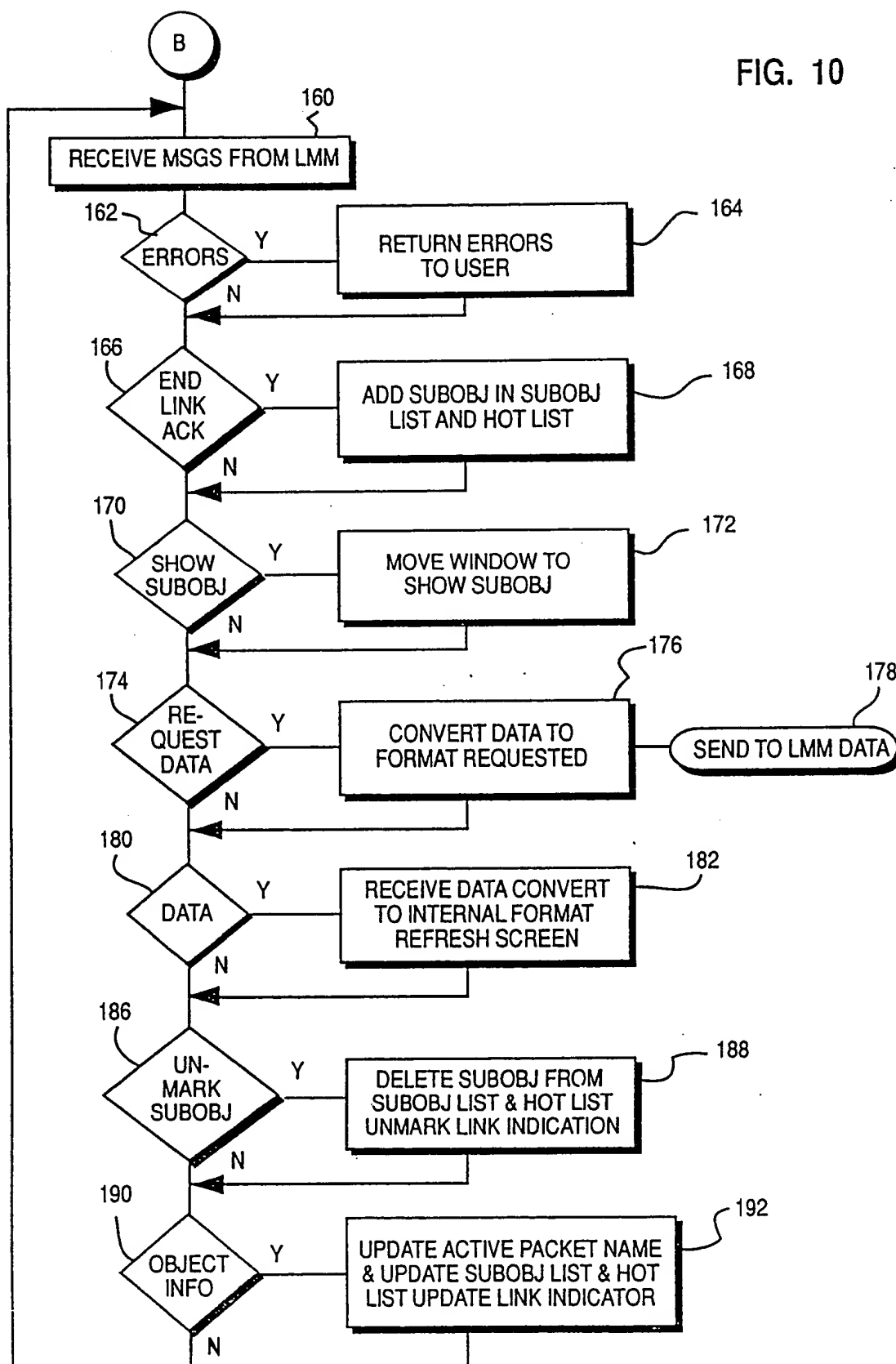
LP APP - HANDLE LINK USER INTERACTION

FIG. 9



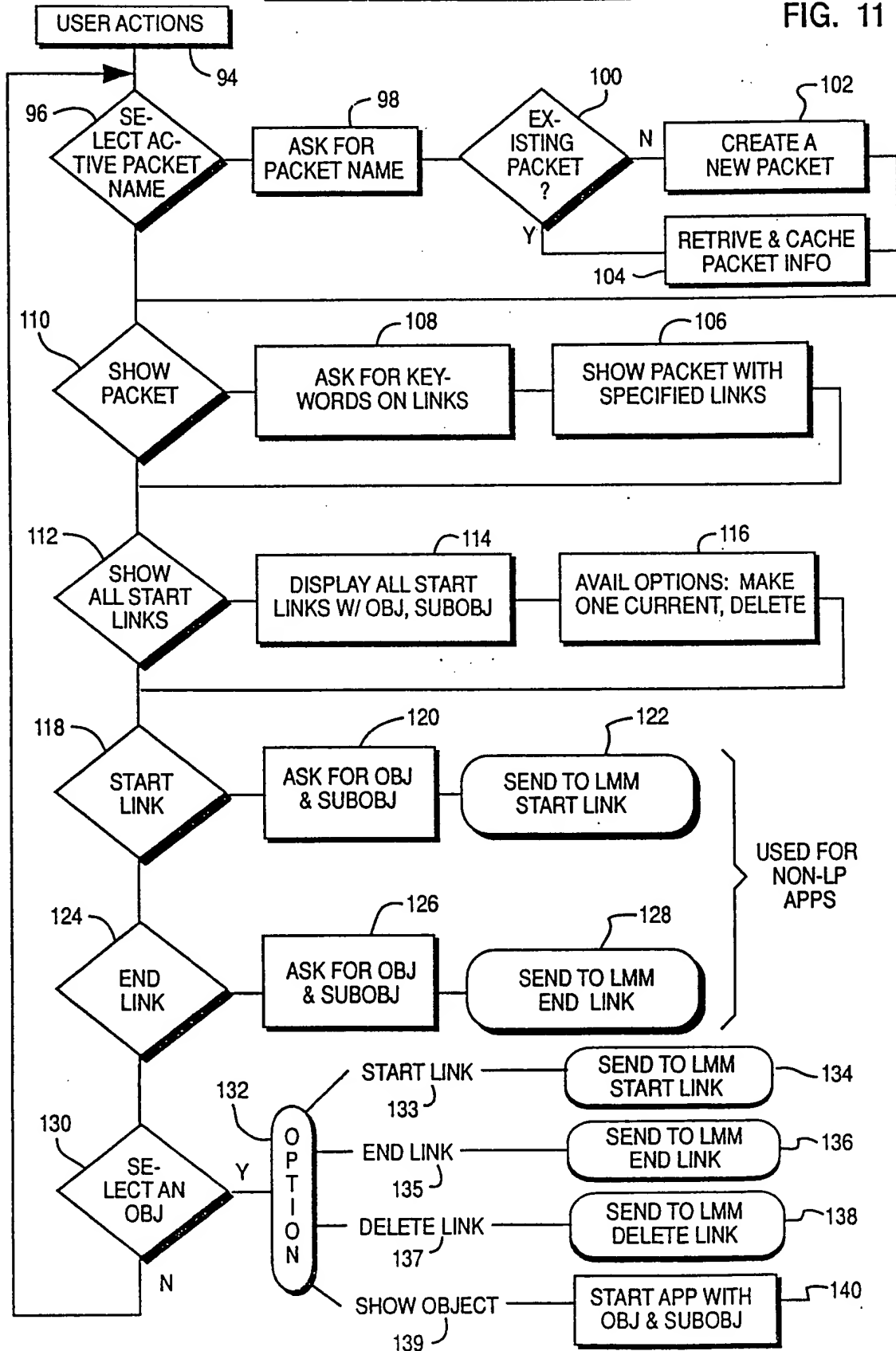
LP APP - HANDLE LINK PROTOCOLS

FIG. 10



PACKET MANAGER FUNCTIONS

FIG. 11



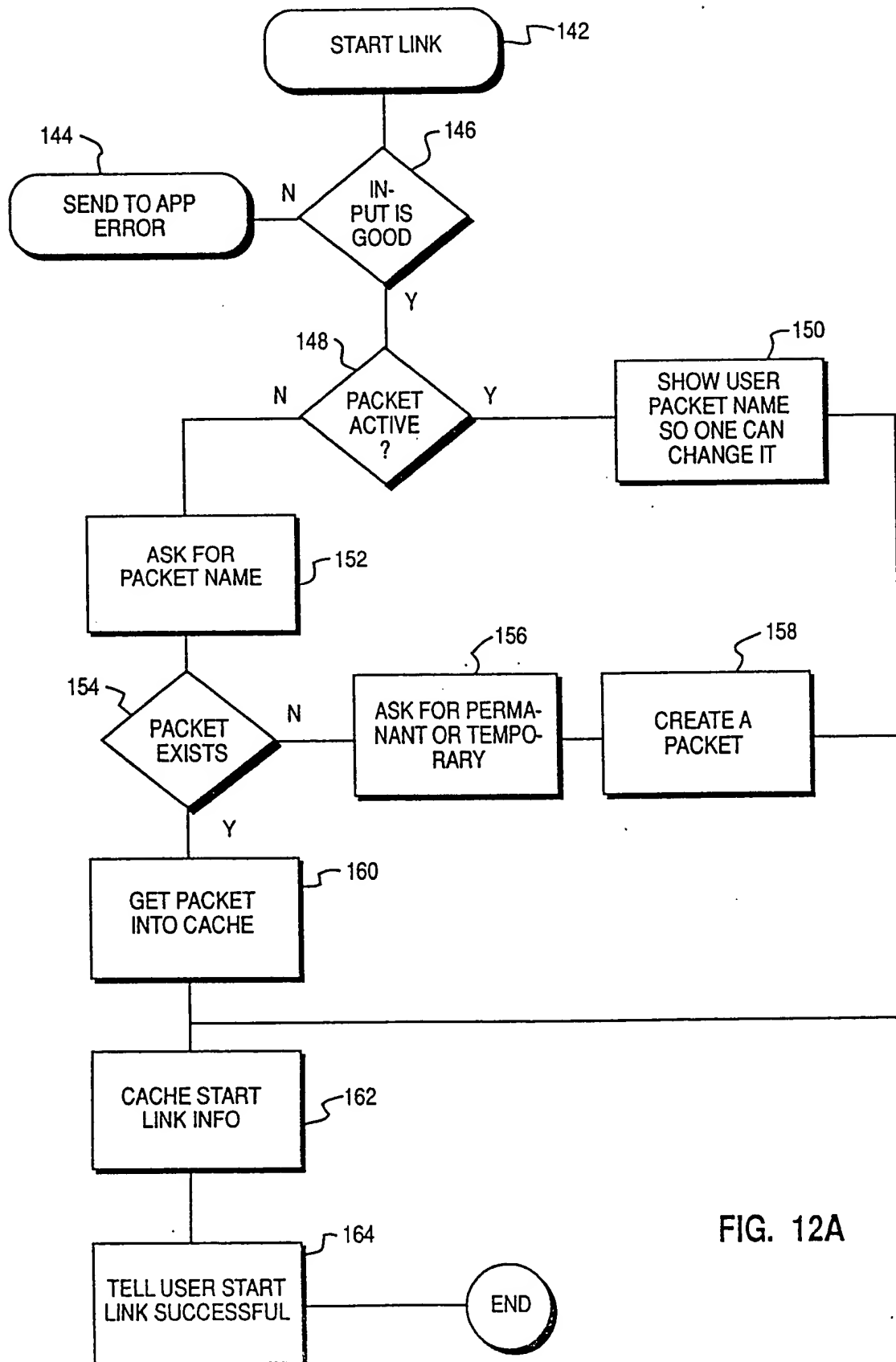
LINK MESSAGE MANAGER FUNCTIONS

FIG. 12A

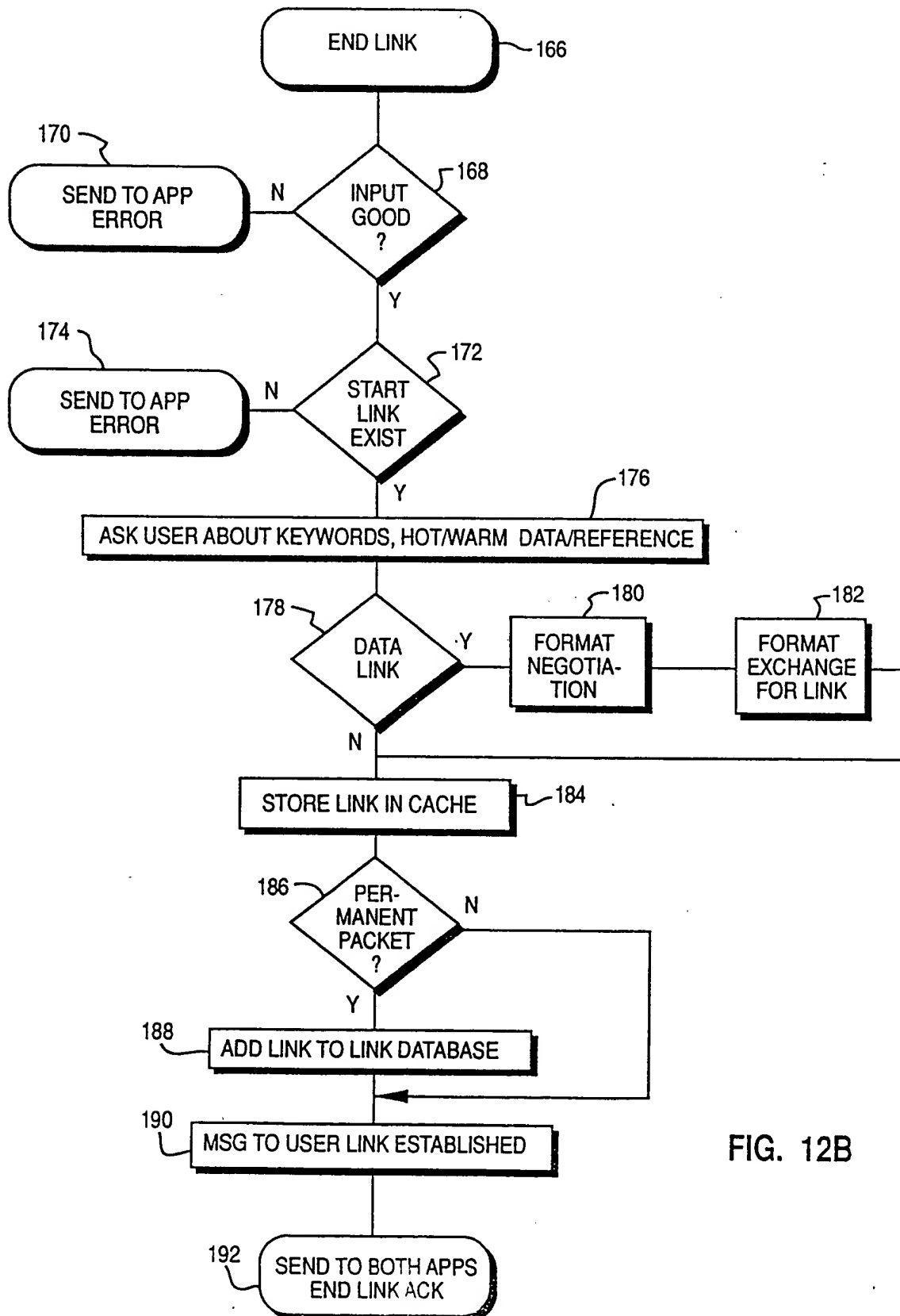
LINK MESSAGE MANAGER FUNCTIONS

FIG. 12B

LINK MESSAGE MANAGER FUNCTIONS

FIG. 12C

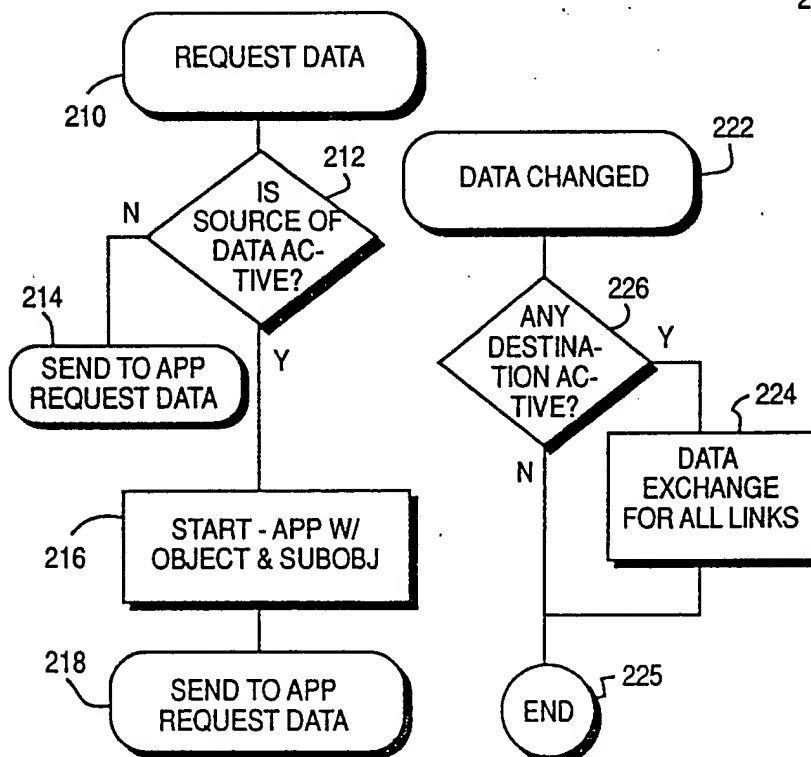
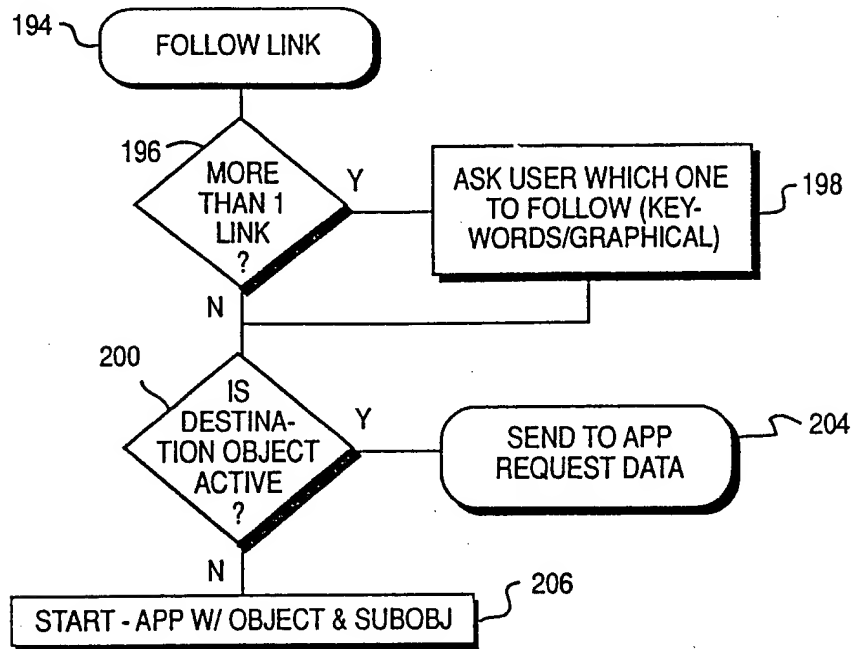


FIG. 12D

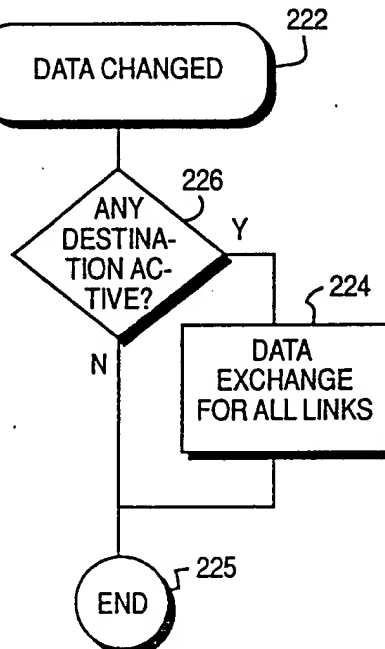


FIG. 12E

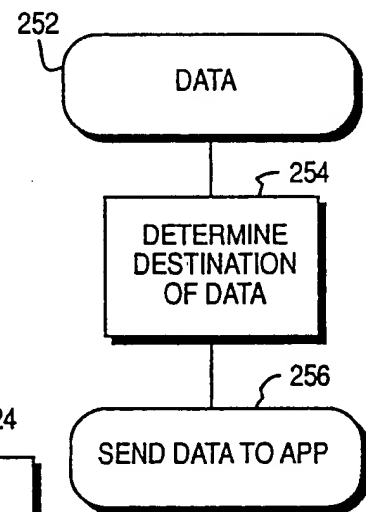


FIG. 12F

LINK MESSAGE MANAGER FUNCTIONS

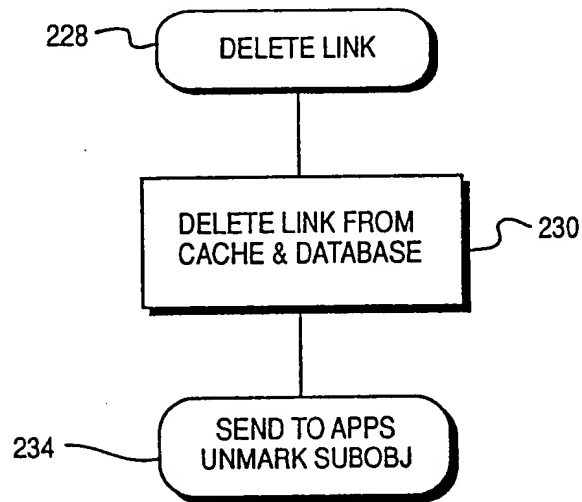


FIG. 12G

DATA EXCHANGE FOR LINK

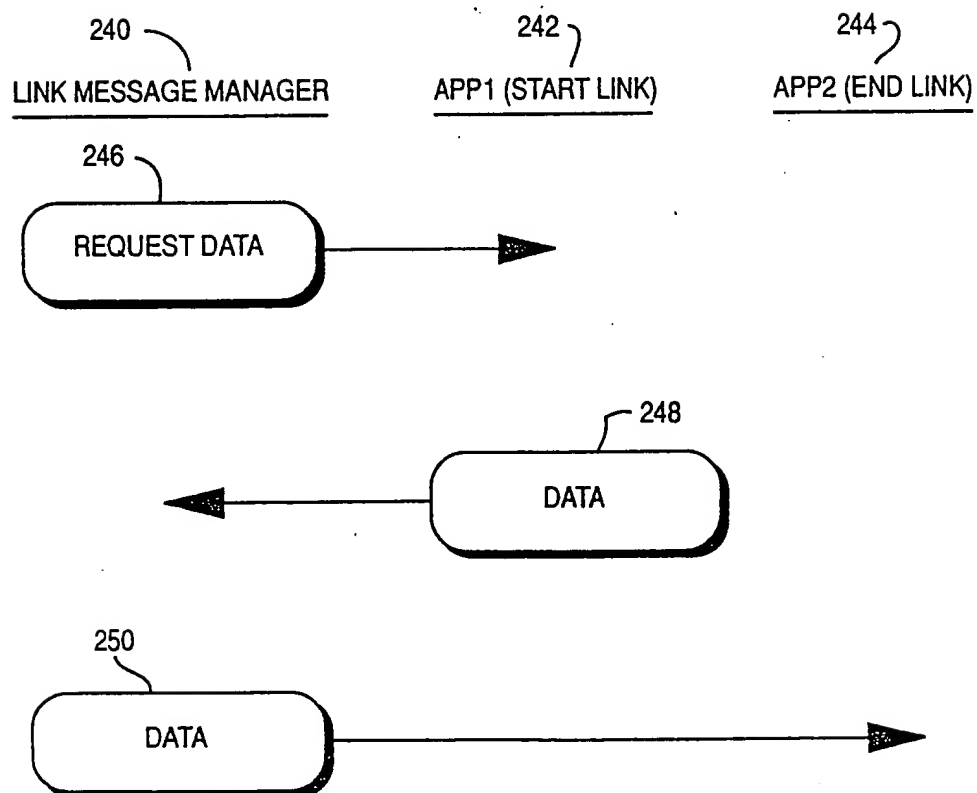


FIG. 13

12

⑤ Int. Cl.⁵: G06F 9/44, G06F 15/40,
G06F 9/46

②② Date of filing: 31.10.89

Armonk, N.Y. 10504(US)

(72) Inventor: Chang, Philip Yen-Tang
6221 Ledge Mountain Drive
Austin Texas 78731(US)

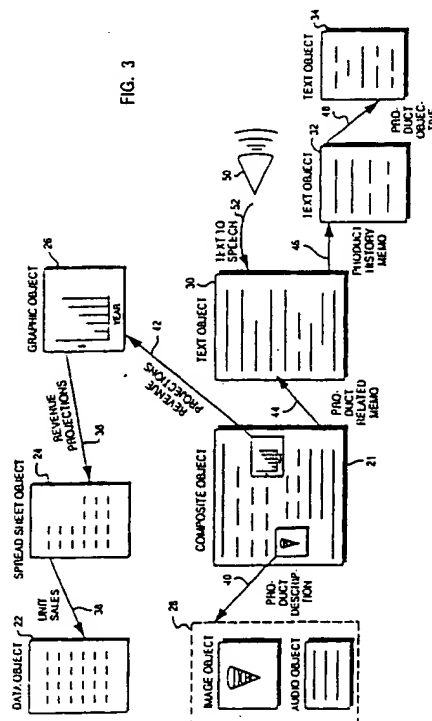
Inventor: Ronga, Gerald Samuel
9902 Richelieu
Austin Texas 78750(US)

74 Representative: Johansson, Lars E. et al
IBM Svenska AB Intellectual Property
Department 4-01
S-163 92 Stockholm(SE)

⑦ Applicant: International Business Machines Corporation
Old Orchard Road

(5d) Interface device and method in a computer system.

57) A linking system for use in a computer environment which supports interapplication messaging and interactive dialog. A link protocol comprised of a set of messages is provided and employed by a link service to interrelate information existing in single or multiple applications. This open protocol facilitates non-joint development of application programs for use in the system. The link service modules provide end user function permitting tasks executed dynamically at run time, of creating or saving links between two objects either of the same or different object types and with or without data exchange, the creation or saving of an application packet comprised of one or more links, refreshing by warm or hot links any linked object, and displaying and navigating links of a packet causing objects to be retrieved and/or refreshed. Function is provided for limited participation in the system by application programs preexisting prior to the protocol.





European Patent
Office

EUROPEAN SEARCH REPORT

Application Number

EP 89 85 0374
Page 2

DOCUMENTS CONSIDERED TO BE RELEVANT			
Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl.5)
X	TECHNIQUE ET SCIENCE INFORMATIQUES. vol. 6, no. 2, 1987, PARIS, FR pages 166 - 169; M. SHAPIRO ET AL.: 'SOS: UN SYSTEME D'EXPLOITATION REPARTI FONDE SUR LES OBJETS' * page 167 - page 169, left column, line 30 * * figures 1,2 *	1	
A	IDEM	2-4, 12, 13, 16, 17, 21, 23	
P, X	EP-A-0 304 071 (WANG LABORATORIES INC.) * the whole document *	1-23	
			TECHNICAL FIELDS SEARCHED (Int. Cl.5)
The present search report has been drawn up for all claims			
Place of search THE HAGUE		Date of completion of the search 25 JUNE 1992	Examiner JONASSON J. T.
CATEGORY OF CITED DOCUMENTS X : particularly relevant if taken alone Y : particularly relevant if combined with another document of the same category A : technological background O : non-written disclosure P : intermediate document T : theory or principle underlying the invention E : earlier patent document, but published on, or after the filing date D : document cited in the application L : document cited for other reasons * : member of the same patent family, corresponding document			